

A 3-CNF-SAT descriptor algebra and the solution of the $P = NP$ conjecture

Prof. Marcel Rémon and Dr. Johan Barthélemy

Abstract

The relationship between the complexity classes P and NP is an unsolved question in the field of theoretical computer science. In this paper, we investigate a descriptor approach based on lattice properties. In a previous paper, we tried to prove that neither $P \neq NP$ nor $P = NP$ was “unprovable” within the a-temporal framework of Mathematics. See [4]. A part of the proof about the impossibility to prove that $P = NP$ turns to be inexact, and yields the first author to investigate deeper into the possibility of $P = NP$.

This paper proposes a new way to decide the satisfiability of any 3-CNF-SAT problem. The analysis of this exact [non heuristical] algorithm shows **a strictly bounded exponential complexity**. The complexity of any 3-CNF-SAT solution is bounded by $O(2^{490})$. This [over-estimated] bound is reached by an algorithm working on the smallest description (via descriptor functions) of the evolving set of solutions in function of the already considered clauses, without exploring these solutions. Any remark about this paper is warmly welcome.

Index Terms

Algorithm Complexity, $P - NP$ problem, 3-CNF-SAT problem

I. THE 3-CNF-SAT PROBLEM

Boolean formulae are built in the usual way from propositional variables x_i and the logical connectives \wedge , \vee and \neg , which are interpreted as conjunction, disjunction, and negation, respectively. A *literal* is a propositional variable or the negation of a propositional variable, and a *clause* is a disjunction of literals. A Boolean formula is *in conjunctive normal form* if and only if it is a conjunction of clauses.

M.Rémon, Department of Mathematics, Namur University, Belgium; marcel.remon@unamur.be
J.Barthélemy, SMART Infrastructure Facilities, University of Wollongong, Australia; johan@uow.edu.au

A *3-CNF formula* φ is a Boolean formula in conjunctive normal form with exactly three literals per clause, like $\varphi := (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) := \psi_1 \wedge \psi_2$. A *3-CNF formula* is composed of n propositional variables x_i and m clauses ψ_j .

The *3-CNF-satisfiability* or *3-CNF-SAT problem* is to decide whether there exists or not logical values for the propositional variables, so that φ can be true. Until now, we do not know whether it is possible or not to check the satisfiability of any given *3-CNF formula* φ in a polynomial time with respect of n , as the *3-CNF-SAT* problem is known to belong to the class NP of problems. See [2] for details.

II. A MATRIX REPRESENTATION OF A 3-CNF FORMULA

A. Definitions

The *size* of a 3-CNF formula φ is defined as the size of the corresponding *Boolean circuit*, i.e. the number of logical connectives in φ . Let us note the following property :

$$size(\varphi) = \mathcal{O}(m) = \mathcal{O}(\alpha \times n) \quad (1)$$

where $\alpha = m/n$ is the *ratio* of clauses with respect to variables. It seems that $\alpha \approx 4.258$ gives the most difficult 3-CNF-SAT problems. See [3].

Let $\varphi(x_1, x_2, \dots, x_n)$ be a 3-CNF formula. The set \mathcal{S}_φ of all *satisfying* solutions is

$$\mathcal{S}_\varphi = \{(x_1, \dots, x_n) \in \{0, 1\}^n \mid \varphi(x_1, \dots, x_n) = 1\} \quad (2)$$

Let $\Sigma_\varphi = \# \mathcal{S}_\varphi$ and $\bar{s}_1, \dots, \bar{s}_{\Sigma_\varphi}$ be the ordered elements of \mathcal{S}_φ . For $1 \leq j \leq \Sigma_\varphi$: $\bar{s}_j = (s_j^1, \dots, s_j^i, \dots, s_j^n)$. We define the *3-CNF-matrix representation* of φ as $[\varphi]$:

$$[\varphi] = \begin{pmatrix} x_1 & x_i & x_n \\ \hline s_1^1 & \dots & s_1^n \\ \vdots & s_j^i & \vdots \\ s_{\Sigma_\varphi}^1 & \dots & s_{\Sigma_\varphi}^n \end{pmatrix} \quad (3)$$

B. Examples

Each clause ψ_i will be represented by a 7×3 matrix. For example,

$$[\psi_1] = [(x_1 \vee x_2 \vee \neg x_3)] = \begin{pmatrix} \begin{array}{ccc|c} x_1 & x_2 & x_3 & \\ \hline 0 & 0 & 0 & \\ 0 & 1 & 0 & \\ 0 & 1 & 1 & \\ 1 & 0 & 0 & \\ 1 & 0 & 1 & \\ 1 & 1 & 0 & \\ 1 & 1 & 1 & \end{array} \end{pmatrix} \quad \text{and} \quad [\psi_2] = [(\neg x_2 \vee x_3 \vee \neg x_4)] = \begin{pmatrix} \begin{array}{ccc|c} x_2 & x_3 & x_4 & \\ \hline 0 & 0 & 0 & \\ 0 & 0 & 1 & \\ 0 & 1 & 0 & \\ 0 & 1 & 1 & \\ 1 & 0 & 0 & \\ 1 & 1 & 0 & \\ 1 & 1 & 1 & \end{array} \end{pmatrix}$$

The 3-CNF formula $\varphi = \psi_1 \wedge \psi_2$ will be represented by a 12×4 matrix :

$$[\varphi] = [(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)] = \begin{pmatrix} \begin{array}{cccc|c} x_1 & x_2 & x_3 & x_4 & \\ \hline 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 1 & \\ 0 & 1 & 0 & 0 & \\ 0 & 1 & 1 & 0 & \\ 0 & 1 & 1 & 1 & \\ 1 & 0 & 0 & 0 & \\ 1 & 0 & 0 & 1 & \\ 1 & 0 & 1 & 0 & \\ 1 & 0 & 1 & 1 & \\ 1 & 1 & 0 & 0 & \\ 1 & 1 & 1 & 0 & \\ 1 & 1 & 1 & 1 & \end{array} \end{pmatrix}$$

This paper defines an algebra on this type of matrices such that $[\varphi] = [\psi_1] \wedge [\psi_2]$.

III. FIRST DEFINITIONS AND PROPERTIES FOR 3-CNF-MATRICES

A. Extension to new variables

Let A be such a matrix, A can be *extended* to new propositional variables by adding columns filled with the neutral sign “.”, meaning that the corresponding variable can be set either to 0 or 1. This new matrix \bar{A} is equivalent to A .

$$A = \begin{pmatrix} \begin{array}{ccc|c} x_1 & x_2 & x_4 & \\ \hline a_1^1 & a_1^2 & a_1^4 & \\ a_j^1 & a_j^2 & a_j^4 & \\ a_{\Sigma_\varphi}^1 & a_{\Sigma_\varphi}^2 & a_{\Sigma_\varphi}^4 & \end{array} \end{pmatrix} \equiv \begin{pmatrix} \begin{array}{ccc|c} x_1 & x_2 & x_3 & x_4 & \\ \hline a_1^1 & a_1^2 & \begin{smallmatrix} 0 \\ 1 \end{smallmatrix} & a_1^4 & \\ a_j^1 & a_j^2 & . & a_j^4 & \\ a_{\Sigma_\varphi}^1 & a_{\Sigma_\varphi}^2 & . & a_{\Sigma_\varphi}^4 & \end{array} \end{pmatrix} = \bar{A} \quad (4)$$

B. Reduction of 3-CNF-matrices

The inverse operation, called *reduction*, replaces two same lines only differing by a 0 and a 1 for a variable, by a unique line with a neutral sign for this variable :

$$A = \left(\begin{array}{ccc|ccc} x_1 & x_2 & x_3 & & & \\ \hline 0 & 0 & 0 & & & \\ 0 & 0 & 1 & & & \\ 0 & 1 & 0 & & & \\ 0 & 1 & 1 & & & \\ 1 & 0 & 0 & & & \\ 1 & 0 & 1 & & & \\ 1 & 1 & 0 & & & \end{array} \right) \equiv \left(\begin{array}{ccc|ccc} x_1 & x_2 & x_3 & & & \\ \hline 0 & 0 & . & & & \\ 0 & 1 & . & & & \\ 1 & 0 & . & & & \\ 1 & 1 & 0 & & & \end{array} \right) \equiv \left(\begin{array}{ccc|ccc} x_1 & x_2 & x_3 & & & \\ \hline 0 & . & . & & & \\ 1 & 0 & . & & & \\ 1 & 1 & 0 & & & \end{array} \right) \quad (5)$$

C. Disjunction of 3-CNF-matrices

Let A and B be two matrices and $\{x_1, \dots, x_n\}$ the union of their support variables. Let \overline{A} and \overline{B} be their extensions over $\{x_1, \dots, x_n\}$. Then we define the *disjunction* of A and B by

$$A \vee B = \left(\begin{array}{ccc|ccc} x_1 & \dots & x_n & & & \\ \hline & \overline{A} & & & & \\ & \overline{B} & & & & \end{array} \right) \quad (6)$$

Of course, this new matrix should be reordered so that the lines are in a ascending order, which can yield sometimes in replacing a line with a neutral sign by two lines with a one and a zero.

D. Block decomposition of 3-CNF-matrices

Let A a matrix such that the *reduction* process yields to lines with neutral sign, then A can be rewritten as the disjunction of smaller matrices. For example,

$$[\psi_1] = \left(\begin{array}{ccc|ccc} x_1 & x_2 & x_3 & & & \\ \hline 0 & 0 & 0 & & & \\ 0 & 1 & 0 & & & \\ 0 & 1 & 1 & & & \\ 1 & 0 & 0 & & & \\ 1 & 0 & 1 & & & \\ 1 & 1 & 0 & & & \\ 1 & 1 & 1 & & & \end{array} \right) = \left(\begin{array}{c|ccc} x_1 & & & \\ \hline 1 & & & \end{array} \right) \vee \left(\begin{array}{cc|cc} x_1 & x_2 & & \\ \hline 0 & 1 & & \end{array} \right) \vee \left(\begin{array}{ccc|ccc} x_1 & x_2 & x_3 & & & \\ \hline 0 & 0 & 0 & & & \end{array} \right)$$

The block decomposition is not unique. For instance, there are 6 different block decompositions for a 3-variables clause.

E. Conjunction of 3-CNF-matrices

Let A and B be two matrices, \bar{A} and \bar{B} their extensions to the joint set of propositional variables. Let \bar{A}_k and \bar{B}_l be the *one line matrices* such that :

$$\bar{A} = \bigvee_{k=1}^{\Sigma_{\bar{A}}} \bar{A}_k \text{ and } \bar{B} = \bigvee_{l=1}^{\Sigma_{\bar{B}}} \bar{B}_l \quad (7)$$

We define the *conjunction* of A and B as

$$A \wedge B \equiv \bar{A} \wedge \bar{B} = \left(\bigvee_{k=1}^{\Sigma_{\bar{A}}} \bar{A}_k \right) \wedge \left(\bigvee_{l=1}^{\Sigma_{\bar{B}}} \bar{B}_l \right) = \bigvee_{k=1}^{\Sigma_{\bar{A}}} \bigvee_{l=1}^{\Sigma_{\bar{B}}} (\bar{A}_k \wedge \bar{B}_l) = \bigvee_{k=1}^{\Sigma_{\bar{A}}} \bigvee_{l=1}^{\Sigma_{\bar{B}}} \bar{C}_{k,l} \quad (8)$$

where

$$\bar{C}_{k,l} = \left(\frac{x_1 \quad x_i \quad x_n}{a_k^1 \quad a_k^i \quad a_k^n} \right) \wedge \left(\frac{x_1 \quad x_i \quad x_n}{b_l^1 \quad b_l^i \quad b_l^n} \right) = \begin{cases} \emptyset & \text{if } \exists c_m^i = "NaN" \\ \left(\frac{x_1 \quad x_i \quad x_n}{c_m^1 \quad c_m^i \quad c_m^n} \right) & \text{otherwise} \end{cases} \quad (9)$$

and

$$c_m^i = \begin{cases} a_k^i & \text{if } a_k^i = b_l^i \\ a_k^i & \text{if } a_k^i \neq b_l^i \text{ and } b_l^i = "." \\ b_l^i & \text{if } a_k^i \neq b_l^i \text{ and } a_k^i = "." \\ "NaN" & \text{otherwise} \end{cases} \quad (10)$$

F. The empty and full 3-CNF-matrices

Let us call \emptyset , the *empty matrix*, with no line at all. The empty matrix is neutral for the disjunction operator \vee and absorbing for the conjunction operator \wedge .

Let us define Ω , the *full matrix*, as a one line matrix with only neutral signs in it. The full matrix is neutral for \wedge and absorbing for \vee .

G. Example of operations

Let us consider the following block decompositions for $[\psi_1]$ and $[\psi_2]$ with x_2 and $(x_2 \ x_3)$ as common supports.

$$[\psi_1] = \begin{pmatrix} \frac{x_1 \ x_2 \ x_3}{0 \ 0 \ 0} \\ 0 \ 1 \ 0 \\ 0 \ 1 \ 1 \\ 1 \ 0 \ 0 \\ 1 \ 0 \ 1 \\ 1 \ 1 \ 0 \\ 1 \ 1 \ 1 \end{pmatrix} = \begin{pmatrix} x_2 \\ 1 \end{pmatrix} \vee \begin{pmatrix} x_2 \ x_3 \\ 0 \ 0 \end{pmatrix} \vee \begin{pmatrix} x_1 \ x_2 \ x_3 \\ 1 \ 0 \ 1 \end{pmatrix}$$

and

$$[\psi_2] = \begin{pmatrix} \frac{x_2 \ x_3 \ x_4}{0 \ 0 \ 0} \\ 0 \ 0 \ 1 \\ 0 \ 1 \ 0 \\ 0 \ 1 \ 1 \\ 1 \ 0 \ 0 \\ 1 \ 1 \ 0 \\ 1 \ 1 \ 1 \end{pmatrix} = \begin{pmatrix} x_2 \\ 0 \end{pmatrix} \vee \begin{pmatrix} x_2 \ x_3 \\ 1 \ 1 \end{pmatrix} \vee \begin{pmatrix} x_2 \ x_3 \ x_4 \\ 1 \ 0 \ 0 \end{pmatrix}$$

$$[\psi_1] \wedge [\psi_2]$$

$$\begin{aligned} &= \emptyset \vee \begin{pmatrix} x_2 \ x_3 \\ 1 \ 1 \end{pmatrix} \vee \begin{pmatrix} x_2 \ x_3 \ x_4 \\ 1 \ 0 \ 0 \end{pmatrix} \vee \begin{pmatrix} x_2 \ x_3 \\ 0 \ 0 \end{pmatrix} \vee \emptyset \vee \emptyset \vee \begin{pmatrix} x_1 \ x_2 \ x_3 \\ 1 \ 0 \ 1 \end{pmatrix} \vee \emptyset \vee \emptyset \\ &= \begin{pmatrix} \frac{x_1 \ x_2 \ x_3 \ x_4}{. \ 0 \ 0 \ .} \\ . \ 1 \ 0 \ 0 \\ . \ 1 \ 1 \ . \\ 1 \ 0 \ 1 \ . \end{pmatrix} \end{aligned}$$

H. Lattice structure of 3-CNF-matrices

A *semi-lattice* (X, \vee) is a pair consisting of a set X and a binary operation \vee which is associative, commutative, and idempotent.

Let us note \mathcal{A} the set of all the 3-CNF-matrices. Then (\mathcal{A}, \vee) and (\mathcal{A}, \wedge) are both semi-lattices, respectively called *join* and *meet* semi-lattices.

Let us define *the two absorption laws* as $x = x \vee (x \wedge y)$ and its dual $x = x \wedge (x \vee y)$. A *lattice* is an algebra (X, \vee, \wedge) satisfying equations expressing associativity, commutativity, and idempotence of \vee and \wedge , and satisfying the two absorption equations.

$(\mathcal{A}, \vee, \wedge)$ is a lattice over the set of 3-CNF-matrices with respect to the disjunction and conjunction operators. Moreover, $(\mathcal{A}, \vee, \wedge)$ is **a distributive bounded lattice** as \wedge is distributive with respect to \vee and $A \vee \Omega = \Omega$ & $A \wedge \emptyset = \emptyset \ \forall A \in \mathcal{A}$. See [1] for more details over lattices.

IV. CHARACTERIZATION THEOREMS VIA FUNCTIONAL DESCRIPTORS

Theorem IV.1: Every non empty 3-CNF-matrix can be characterized by a one-line parameterized 3-CNF-matrix, called its functional matrix description.

$$\vee [\varphi] = \left(\begin{array}{ccc} x_1 & x_i & x_n \\ s_1^1 & \cdots & s_1^n \\ \vdots & s_j^i & \vdots \\ s_{\Sigma_\varphi}^1 & \cdots & s_{\Sigma_\varphi}^n \end{array} \right) \neq \emptyset, \exists n \text{ functions } f_i : \{0,1\}^i \rightarrow \{0,1\} \text{ such that}$$

$$[\varphi] = \bigvee_{(\alpha_1, \dots, \alpha_n) \in \{0,1\}^n} \left(\begin{array}{cccccc} x_1 & \cdots & x_i & \cdots & x_n \\ f_1(\alpha_1) & \cdots & f_i(\alpha_1, \dots, \alpha_i) & \cdots & f_n(\alpha_1, \dots, \alpha_n) \end{array} \right) \quad (11)$$

$$\stackrel{\text{notation}}{\equiv} \left[\begin{array}{c} f_1(\alpha_1) \\ \vdots \\ f_n(\alpha_1, \dots, \alpha_n) \end{array} \right] \quad (12)$$

So, the knowledge of $f_1(\alpha_1), \dots, f_i(\alpha_1, \dots, \alpha_i), \dots, f_n(\alpha_1, \dots, \alpha_n)$ characterizes fully $[\varphi]$.

These modulo-2 functions are called **the functional descriptors** of φ .

Example :

$$\begin{aligned} [\varphi] &= [(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)] \\ &= \bigvee_{(\alpha_1, \dots, \alpha_4) \in \{0,1\}^4} \left(\frac{x_1 \quad x_2 \quad x_3 \quad x_4}{\alpha_1 \quad \alpha_2 \quad (\alpha_1 + 1)(\alpha_2 + 1)\alpha_3 + \alpha_3 \quad \alpha_2(\alpha_3 + 1)\alpha_4 + \alpha_4} \right) \pmod{2} \end{aligned}$$

Proof:

- The theorem is satisfied for $n = 1$ as

$$\left(\frac{x_1}{1} \right) = \left(\frac{x_1}{f_1(\alpha_1) \equiv 1} \right) ; \left(\frac{x_1}{0} \right) = \left(\frac{x_1}{f_1(\alpha_1) \equiv 0} \right) ; \left(\frac{x_1}{0 \atop 1} \right) = \bigvee_{\alpha_1 \in \{0,1\}} \left(\frac{x_1}{\alpha_1} \right)$$

- Let the theorem be true for $n - 1$ and $[\varphi]$ be a 3-CNF-matrix of dimension n . There exist two 3-CNF-matrices $[\varphi_1]$ and $[\varphi_2]$ of size $n - 1$ such that :

$$[\varphi] = \bigvee_{\alpha_i \in \{0,1\}} \left(\frac{x_1 \quad x_2 \cdots x_n}{0 \quad f_2(\alpha_2) \cdots f_n(\alpha_2, \dots, \alpha_n)} \right) \bigvee_{\alpha_i \in \{0,1\}} \left(\frac{x_1 \quad x_2 \cdots x_n}{1 \quad g_2(\alpha_2) \cdots g_n(\alpha_2, \dots, \alpha_n)} \right)$$

$$\text{Thus} \quad [\varphi] = \bigvee_{\alpha_i \in \{0,1\}} \left(\frac{x_1 \cdots x_n}{h_1(\alpha_1) \cdots h_n(\alpha_1, \dots, \alpha_n)} \right)$$

where

$$h_1(\alpha_1) = \alpha_1$$

$$h_i(\alpha_1, \dots, \alpha_i) = (\alpha_1 + 1)f_i(\alpha_2, \dots, \alpha_i) + \alpha_1 g_i(\alpha_2, \dots, \alpha_i) \pmod{2} \quad \text{for } i \neq 1$$

■

Corollary IV.1: The functional descriptors of φ are modulo-2 multi-linear combinations of α_i .

Proof: This is a mere consequence of the recursive definition of $h_i(\alpha_1, \dots, \alpha_i)$.

$$\text{So, } h_i(\alpha_1, \dots, \alpha_i) = \sum_{(\delta_1, \dots, \delta_i) \in \{0,1\}^i} \Delta_i(\delta_1, \dots, \delta_i) \alpha_1^{\delta_1} \cdots \alpha_i^{\delta_i} \pmod{2} \quad (13)$$

where $\Delta_i(\delta_1, \dots, \delta_i) \in \{0,1\}$

$\Delta_i(\delta_1, \dots, \delta_i)$ is called the signature of $h_i(\alpha_1, \dots, \alpha_i)$.

■

Example :

Consider a clause $\psi \equiv [\neg]x_r \vee [\neg]x_s \vee [\neg]x_t$ where $1 \leq r < s < t \leq n$. $[\psi]$ is then characterized by the following characterization functions :

$$h_i(\alpha_1, \dots, \alpha_i) = \alpha_i \quad \forall i < t$$

$$h_t(\alpha_r, \alpha_s, \alpha_t) = \begin{cases} (\alpha_r + 1)(\alpha_s + 1)(\alpha_t + 1) + \alpha_t & \text{if } \psi = x_r \vee x_s \vee x_t \\ (\alpha_r + 1)(\alpha_s + 1) \alpha_t + \alpha_t & \text{if } \psi = x_r \vee x_s \vee \neg x_t \\ (\alpha_r + 1) \alpha_s (\alpha_t + 1) + \alpha_t & \text{if } \psi = x_r \vee \neg x_s \vee x_t \\ (\alpha_r + 1) \alpha_s \alpha_t + \alpha_t & \text{if } \psi = x_r \vee \neg x_s \vee \neg x_t \\ \alpha_r (\alpha_s + 1)(\alpha_t + 1) + \alpha_t & \text{if } \psi = \neg x_r \vee x_s \vee x_t \\ \alpha_r (\alpha_s + 1) \alpha_t + \alpha_t & \text{if } \psi = \neg x_r \vee x_s \vee \neg x_t \\ \alpha_r \alpha_s (\alpha_t + 1) + \alpha_t & \text{if } \psi = \neg x_r \vee \neg x_s \vee x_t \\ \alpha_r \alpha_s \alpha_t + \alpha_t & \text{if } \psi = \neg x_r \vee \neg x_s \vee \neg x_t \end{cases} \quad (14)$$

Theorem IV.2: The conjunction operator \wedge between two sets of clauses can be rewritten as the merging of their characterization functions : $[h_i(\cdot)] = [f_i(\cdot)] \wedge [g_i(\cdot)]$.

$$\text{Let } [\varphi] \equiv \begin{bmatrix} f_1(\alpha_1) \\ \vdots \\ f_n(\alpha_1, \dots, \alpha_n) \end{bmatrix} \text{ and } [\varphi'] \equiv \begin{bmatrix} g_1(\alpha_1) \\ \vdots \\ g_n(\alpha_1, \dots, \alpha_n) \end{bmatrix}$$

Note : φ or φ' should be extended if necessary in order to get the same support of propositional variables. Remember that all operations are *modulo 2* : $\alpha_i + \alpha_i = 0$, $\alpha_i^2 = \alpha_i$ and $(\alpha_i + 1)\alpha_i = 0$ for all α_i .

$$\text{Then } [\varphi] \wedge [\varphi'] \equiv \begin{bmatrix} h_1(\alpha_1) \\ \vdots \\ h_n(\alpha_1, \dots, \alpha_n) \end{bmatrix}$$

where for $1 \leq t \leq n$:

$$\begin{aligned}
 h_t(\alpha_1, \dots, \alpha_t) = & (\alpha_t + 1) \cdot \{ [f_t(\alpha_1, \dots, \alpha_{t-1}, 0) + g_t(\alpha_1, \dots, \alpha_{t-1}, 0)] \\
 & \cdot [f_t(\alpha_1, \dots, \alpha_{t-1}, 1) \cdot g_t(\alpha_1, \dots, \alpha_{t-1}, 1)] \\
 & + [f_t(\alpha_1, \dots, \alpha_{t-1}, 0) \cdot g_t(\alpha_1, \dots, \alpha_{t-1}, 0)] \} \\
 + & \alpha_t \cdot \{ [f_t(\alpha_1, \dots, \alpha_{t-1}, 1) + g_t(\alpha_1, \dots, \alpha_{t-1}, 1)] \\
 & \cdot [f_t(\alpha_1, \dots, \alpha_{t-1}, 0) + g_t(\alpha_1, \dots, \alpha_{t-1}, 0)] \\
 & + [f_t(\alpha_1, \dots, \alpha_{t-1}, 1) + g_t(\alpha_1, \dots, \alpha_{t-1}, 1)] \\
 & \cdot [f_t(\alpha_1, \dots, \alpha_{t-1}, 0) \cdot g_t(\alpha_1, \dots, \alpha_{t-1}, 0)] \\
 & + [f_t(\alpha_1, \dots, \alpha_{t-1}, 1) \cdot g_t(\alpha_1, \dots, \alpha_{t-1}, 1)] \} \pmod{2}
 \end{aligned} \tag{15}$$

Moreover if there exists a (unique) $j < t$, related to the highest α_j such that :

$$\begin{aligned}
 g_j^*(\alpha_1, \dots, \alpha_j) \equiv & [f_t(\alpha_1, \dots, \alpha_{t-1}, 0) + g_t(\alpha_1, \dots, \alpha_{t-1}, 0)] \cdot \\
 & [f_t(\alpha_1, \dots, \alpha_{t-1}, 1) + g_t(\alpha_1, \dots, \alpha_{t-1}, 1)] \neq 0
 \end{aligned} \tag{16}$$

[An additional constraint over α_j induced by the conjunction operation]

\Rightarrow call to a new merging $f_j(\alpha_1, \dots, \alpha_j) \wedge g_j^*(\alpha_1, \dots, \alpha_j)$

using a recursive call to definition (15).

(17)

Recursivity will end as soon as there is no longer such $g_j^*(\alpha_1, \dots, \alpha_j) = 1$ or when $g_j^*(\alpha_1, \dots, \alpha_j)$

is no longer a function of α_i but a constant always equal to 1.

Proof: Consider the possible values for $f_t(\alpha_1, \dots, \alpha_t)$ and $g_t(\alpha_1, \dots, \alpha_t)$ in equation (15)

when $\alpha_t \in \{0, 1\}$:

$$\bullet f_t(\alpha_1, \dots, \alpha_t) = g_t(\alpha_1, \dots, \alpha_t) \text{ for } \alpha_t \in \{0, 1\}$$

$$\Downarrow$$

$$\begin{aligned} h_t(\alpha_1, \dots, \alpha_t) &= (\alpha_t + 1) \cdot \{ [f_t(\cdot, 0) + g_t(\cdot, 0)] \cdot [f_t(\cdot, 1) \cdot g_t(\cdot, 1)] + [f_t(\cdot, 0) \cdot g_t(\cdot, 0)] \} + \\ &\quad \alpha_t \cdot \{ [f_t(\cdot, 1) + g_t(\cdot, 1)] \cdot [f_t(\cdot, 0) + g_t(\cdot, 0)] + \\ &\quad [f_t(\cdot, 1) + g_t(\cdot, 1)] \cdot [f_t(\cdot, 0) \cdot g_t(\cdot, 0)] + [f_t(\cdot, 1) \cdot g_t(\cdot, 1)] \} \\ &= (\alpha_t + 1) \cdot f_t(\cdot, 0) + \alpha_t \cdot f_t(\cdot, 1) \text{ [as } f_t(\cdot) + g_t(\cdot) = 0 \text{ and } f_t(\cdot) \cdot g_t(\cdot) = f_t^2(\cdot) = f_t(\cdot)]} \\ &= f_t(\alpha_1, \dots, \alpha_t) = h_t(\alpha_1, \dots, \alpha_t) \\ &\quad [h_t() \text{ is thus the conjunction of } f_t() \text{ and } g_t()] \end{aligned}$$

$$\bullet f_t(\alpha_1, \dots, 0) = g_t(\alpha_1, \dots, 0) \text{ but } f_t(\alpha_1, \dots, 1) \neq g_t(\alpha_1, \dots, 1)$$

$$\Downarrow$$

$$\begin{aligned} h_t(\alpha_1, \dots, \alpha_t) &= (\alpha_t + 1) \cdot f_t(\cdot, 0) + \alpha_t \cdot f_t(\cdot, 1) \text{ [as } f_t(\cdot, 1) + g_t(\cdot, 1) = 1 \text{ and } f_t(\cdot, 1) \cdot g_t(\cdot, 1) = 0] \\ &= f_t(\alpha_1, \dots, 0) = g_t(\alpha_1, \dots, 0) \\ &\quad [h_t() \text{ sends } \alpha_t \text{ to the value where } f_t() = g_t()] \end{aligned}$$

$$\bullet f_t(\alpha_1, \dots, 1) = g_t(\alpha_1, \dots, 1) \text{ but } f_t(\alpha_1, \dots, 0) \neq g_t(\alpha_1, \dots, 0)$$

$$\Downarrow$$

$$\begin{aligned} h_t(\alpha_1, \dots, \alpha_t) &= (\alpha_t + 1) \cdot f_t(\cdot, 1) + \alpha_t \cdot f_t(\cdot, 0) \text{ [as } f_t(\cdot, 0) + g_t(\cdot, 0) = 1 ; f_t(\cdot, 0) \cdot g_t(\cdot, 0) = 0] \\ &= f_t(\alpha_1, \dots, 1) = g_t(\alpha_1, \dots, 1) \\ &\quad [h_t() \text{ sends } \alpha_t \text{ to the value where } f_t() = g_t()] \end{aligned}$$

$$\begin{aligned}
& \bullet f_t(\alpha_1, \dots, 1) \neq h_t(\alpha_1, \dots, 1) \text{ and } f_t(\alpha_1, \dots, 0) \neq g_t(\alpha_1, \dots, 0) \\
& \Downarrow \text{ [Impossibility to merge the two functions } f_t() \text{ and } g_t() \text{]} \\
& \Downarrow \text{ [No constraint over } \alpha_t \text{ but a induced constraint over some } \alpha_j, j < t \text{]} \\
& h_t(\alpha_1, \dots, \alpha_t) = \alpha_t \text{ [as } f_t() + g_t() = 1 \text{ and } f_t() \cdot g_t() = 0 \text{]} \\
& \text{but } [f_t(\cdot, 0) + g_t(\cdot, 0)] \cdot [f_t(\cdot, 1) + g_t(\cdot, 1)] = \text{fonction}(\alpha_1, \dots, \alpha_j) = 1 \\
& \text{and } g_j^*(\alpha_1, \dots, \alpha_j) := [f_t(\cdot, 0) + g_t(\cdot, 0)] \cdot [f_t(\cdot, 1) + g_t(\cdot, 1)] + g_j(\alpha_1, \dots, \alpha_j) = 1 \\
& \text{[New additional constraint over } \alpha_j, j < t, \text{ so that the impossibility} \\
& \text{cannot occur anymore, as } g_j(\cdot, \alpha_j) \rightarrow g_j(\cdot, \alpha_j) + 1 \text{ when it appears.]}
\end{aligned}$$

■

Example :

Consider the following sets of clauses :

- $\varphi = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4)$
- $\varphi' = (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3 \vee \neg x_5) \wedge (\neg x_1 \vee x_3 \vee \neg x_5)$

Then

$$\begin{aligned}
[\varphi] &= \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_1\alpha_3 + \alpha_2\alpha_3 + \alpha_1\alpha_2\alpha_3 \\ \alpha_4 \\ \alpha_5 \end{bmatrix} \wedge \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 + \alpha_2\alpha_4 + \alpha_2\alpha_3\alpha_4 \\ \alpha_5 \end{bmatrix} \wedge \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 + \alpha_1\alpha_4 + \alpha_1\alpha_3\alpha_4 \\ \alpha_5 \end{bmatrix} \\
&= \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_1\alpha_3 + \alpha_2\alpha_3 + \alpha_1\alpha_2\alpha_3 \\ \alpha_4 + \alpha_1\alpha_4 + \alpha_2\alpha_4 + \alpha_1\alpha_2\alpha_4 + \alpha_1\alpha_3\alpha_4 + \alpha_2\alpha_3\alpha_4 + \alpha_1\alpha_2\alpha_3\alpha_4 \\ \alpha_5 \end{bmatrix}
\end{aligned}$$

$$\begin{aligned}
[\varphi'] &= \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 + \alpha_1\alpha_3 + \alpha_1\alpha_2\alpha_3 \\ \alpha_4 \\ \alpha_5 \end{bmatrix} \wedge \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 + \alpha_2\alpha_5 + \alpha_2\alpha_3\alpha_5 \end{bmatrix} \wedge \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 + \alpha_1\alpha_5 + \alpha_1\alpha_3\alpha_5 \end{bmatrix} \\
&= \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 + \alpha_1\alpha_3 + \alpha_1\alpha_2\alpha_3 \\ \alpha_4 \\ \alpha_5 + \alpha_1\alpha_5 + \alpha_2\alpha_5 + \alpha_1\alpha_2\alpha_5 + \alpha_2\alpha_3\alpha_5 \end{bmatrix}
\end{aligned}$$

And

$$[\varphi] \wedge [\varphi'] = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_2\alpha_3 \\ \alpha_4 + \alpha_1\alpha_4 + \alpha_2\alpha_4 + \alpha_1\alpha_2\alpha_4 + \alpha_1\alpha_3\alpha_4 + \alpha_2\alpha_3\alpha_4 + \alpha_1\alpha_2\alpha_3\alpha_4 \\ \alpha_5 + \alpha_1\alpha_5 + \alpha_2\alpha_5 + \alpha_1\alpha_2\alpha_5 + \alpha_2\alpha_3\alpha_5 \end{bmatrix}$$

In this example, no recursive call is done. We get $h_3(\cdot) := (\alpha_1\alpha_3 + \alpha_2\alpha_3 + \alpha_1\alpha_2\alpha_3) \wedge (\alpha_3 + \alpha_1\alpha_3 + \alpha_1\alpha_2\alpha_3) \stackrel{(15)}{\Rightarrow} h_3(\cdot) := (\alpha_3 + 1) \cdot \{0 \cdot \alpha_2 + 0\} + \alpha_3 \cdot \{(\alpha_2 + 1) \cdot 0 + (\alpha_2 + 1) \cdot 0 + \alpha_2\} = \alpha_2\alpha_3$, as $f_3(\alpha_1, \alpha_2, 0) = 0$, $f_3(\alpha_1, \alpha_2, 1) = \alpha_1 + \alpha_2 + \alpha_1\alpha_2$, $g_3(\alpha_1, \alpha_2, 0) = 0$, $g_3(\alpha_1, \alpha_2, 1) = 1 + \alpha_1 + \alpha_1\alpha_2$, $f_3(\alpha_1, \alpha_2, 1) + g_3(\alpha_1, \alpha_2, 1) = \alpha_2 + 1$ and $f_3(\alpha_1, \alpha_2, 1) \cdot g_3(\alpha_1, \alpha_2, 1) = \alpha_2$.

V. AN APPROACH TO THE 3-CNF-SAT PROBLEM VIA DESCRIPTORS

A. Boolean descriptors

The usual presentation of a 3-CNF-SAT problem consists of a list on m 3-CNF clauses defined over n propositional variables. These m clauses describe perfectly the set of solutions for the 3-CNF-SAT problem and can be considered as **Boolean descriptors** of the 3-CNF-SAT problem. These Boolean descriptors are of **linear complexity** as they can be represented by an array of dimension $3 \times m$.

The difficulty with such Boolean descriptors is that there is **no simple or direct relation** between them and the set of solutions or the answer to the satisfiability question.

B. 3-CNF-matrix descriptors

This paper proposes in (3) a 3-CNF-matrix description of a 3-CNF-SAT problem. These descriptors (each line in the 3-CNF-matrix) can be of **exponential complexity** as there are as many descriptors as solutions. Even if one uses the reduction version of the 3-CNF-matrix description as explained in (5), simulations show that the complexity remains exponential. The interest of these descriptors is the **direct link** between them and the set of solutions or the answer to the satisfiability question.

C. Functional descriptors

This paper proposes also in (11) a 3-CNF-matrix **functional description** for any 3-CNF-SAT problem. These functional descriptors are of **unknown complexity**, at least at this stage of the paper.

These functional descriptors are *somehow in between* both previous types of descriptors, as they are in an **exponential relation** to the set of solutions and in an **direct relation** with the satisfiability question. Indeed, given the functional descriptors, it is straightforward to give the answer to the satisfiability question : no if the functional descriptors does not exist, and yes otherwise. However, one needs to generate all possible values for α_i to get the entire set of solutions, and that can take an exponential time.

Conclusion : *The approach of the 3-CNF-SAT problem via functional descriptors seems to be promising as it does not consider the set of all solutions, but only focuses on the sole question about satisfiability.*

VI. COMPLEXITY ANALYSIS OF THE FUNCTIONAL DESCRIPTOR APPROACH

A. A first measure of the complexity for functional descriptors

Theorem VI.1: The complexity of the functional descriptor approach for a 3-CNF-SAT problem with m clauses and n propositional variables is

$$\mathcal{O}(m n^2 \max_{1 \leq t \leq n} \max_{1 \leq j \leq m} \text{len}_j(h_t))$$

where $\text{len}_j(h_t)$ is the number of terms in $h_t(\cdot)$ when the j first clauses are considered :

$$\text{Let } \text{len}(h_t) \equiv \sum_{(\delta_1, \dots, \delta_t) \in \{0,1\}^t} \Delta_t(\delta_1, \dots, \delta_t) \text{ [see (13) for the definition of } \Delta_t \text{]} \quad (18)$$

So $\text{len}_j(h_t) = \text{len}(h_t)$ at stage j of the computations.

Proof: Let us compute the complexity of $f_t(\cdot) \wedge g_t(\cdot)$ in (15). First of all, one has to compute the four functions in square brackets : $[f_t(\cdot, 0) + g_t(\cdot, 0)]$, $[f_t(\cdot, 1) + g_t(\cdot, 1)]$, $[f_t(\cdot, 0) \cdot g_t(\cdot, 0)]$ and $[f_t(\cdot, 1) \cdot g_t(\cdot, 1)]$. We have :

$$\text{len}(f_t(\cdot, 0)) \leq \text{len}(f_t(\cdot, \alpha_t)) \text{ and } \text{len}(f_t(\cdot, 1)) \leq \text{len}(f_t(\cdot, \alpha_t)) [\equiv \text{len}(f_t)]$$

$$\text{len}(g_t(\cdot, 0)) \leq \text{len}(g_t(\cdot, \alpha_t)) \text{ and } \text{len}(g_t(\cdot, 1)) \leq \text{len}(g_t(\cdot, \alpha_t)) [\equiv \text{len}(g_t)]$$

$$\text{len}(f_t + g_t) \leq \text{len}(f_t) + \text{len}(g_t) \leq \text{len}(f_t) \cdot \text{len}(g_t) \text{ when } \text{len}(f_t) > 2 \text{ and } \text{len}(g_t) > 2$$

The complexity for the four functions is then $\mathcal{O}(\text{len}(f_t) \cdot \text{len}(g_t))$.

The complexity for computing $h_t(\cdot)$ in (15) is :

$$\begin{aligned} & \mathcal{O}(3 \cdot [(\text{len}(f_t) \cdot \text{len}(g_t))^2 + (\text{len}(f_t) \cdot \text{len}(g_t))] + 2 \cdot [2(\text{len}(f_t) \cdot \text{len}(g_t))^2 + (\text{len}(f_t) \cdot \text{len}(g_t))]) \\ &= \mathcal{O}(7 \cdot (\text{len}(f_t) \cdot \text{len}(g_t))^2 + 5 \cdot (\text{len}(f_t) \cdot \text{len}(g_t))) \\ &= \mathcal{O}[(\text{len}(f_t) \cdot \text{len}(g_t))^2] \text{ for large } \text{len}(f_t) \cdot \text{len}(g_t) \end{aligned} \quad (19)$$

Note : it needs three runs over the formula in the brackets to do the product with $(\alpha_t + 1)$, one to compute the formula, one to multiply it by α_t and one to add both results. Similarly, it takes two runs to compute the product with α_t .

Using the same argumentation, we have :

$$\text{len}(h_t) = \mathcal{O}([\text{len}(f_t) \cdot \text{len}(g_t)]^2) \quad \text{for large } \text{len}(f_t) \cdot \text{len}(g_t) \quad (20)$$

and for the recursive call with $j < t$ [see (16)]

$$\text{len}(g_j^*) = \mathcal{O}([\text{len}(f_t) \cdot \text{len}(g_t)]^2) \quad \text{for large } \text{len}(f_t) \cdot \text{len}(g_t) \quad (21)$$

To solve the 3-CNF-SAT problem, one should compute all n functional descriptors $h_t(\cdot)$, each of them with at most n recursive calls, and this for each step of integration of the m clauses. So, using the equivalence between (20) and (19), the overall complexity of the functional approach to 3-CNF-SAT problem will be of order $\mathcal{O}(m n^2 \max_{1 \leq t \leq n} \max_{1 \leq j \leq m} \text{len}_j(h_t))$. ■

B. Non uniformly distributed versus uniformly distributed literals in 3-CNF-SAT problems

Theorem VI.2: The most difficult 3-CNF-SAT problems are uniformly distributed ones.

Note : The invariance structure of 3-CNF-SAT problem is important with respect to the complexity of the functional descriptor approach. It is then normal that problems with uniformly distributed propositional variables are harder as no re-labeling of the variables can be done to reduce the complexity. A simple example of the importance of re-labeling is proposed just after the following proof.

Proof:

• Negative and positive literals

First, let us note that the computations involving negative literals are easier and faster than for positive ones. This is a mere consequence of our definition for $h_t(\cdot)$ in (14). So *the most difficult problems will be the balanced one with respect to the proportion of negative and positive literals*. Otherwise, we inverse some variables in order to get the maximum of negative literals. Let us suppose from here that the proportion of positive and negative literals is quasi equal for each variable.

• Some definitions

Let us divide now the clauses in two sets. The first set contains all the clauses with the

higher indexed literal being positive and the second with the negative ones :

$$\begin{aligned}
Cl^+ &= \bigcup_t Cl^+(x_t) \\
&= \bigcup_t \{\psi_i := [\neg]x_r \vee [\neg]x_s \vee x_t \text{ with } r < s < t, \text{ or any permutation of } x_r, x_s, x_t\} \\
Cl^- &= \bigcup_t Cl^-(x_t) \\
&= \bigcup_t \{\psi_i := [\neg]x_r \vee [\neg]x_s \vee \neg x_t \text{ with } r < s < t, \text{ or any permutation of } x_r, x_s, x_t\}
\end{aligned}$$

$$\text{and } V^+(x_t) = \{x_i \ (i < t) \mid \exists \psi \in Cl^+(x_t) : x_i \text{ appears in } \psi\}$$

$$V^-(x_t) = \{x_i \ (i < t) \mid \exists \psi \in Cl^-(x_t) : x_i \text{ appears in } \psi\}$$

$$V(x_t) = V^+(x_t) \cup V^-(x_t)$$

By construction, there is at least one solution for each x_t when considering clauses only in $Cl^+[x_t = 1]$ or in $Cl^-[x_t = 0]$. Moreover, the computation of the functional descriptors will not involved recursive calls [see (16)] as no impossibility exists for any x_t .

As $h_t(\cdot)$ is a multi-linear combination of the α_i corresponding to the literals x_i ($i \leq t$) found in clauses where x_t has the highest index, there will be at most $2^{(\#V^+(x_t) + 1)}$ [respectively $2^{(\#V^-(x_t) + 1)}$] terms in $h_t(\cdot)$ computed over Cl^+ [resp. Cl^-]. So $\text{len}(h_t)_{|Cl^+} \leq 2^{(\#V^+(x_t) + 1)}$ and $\text{len}(h_t)_{|Cl^-} \leq 2^{(\#V^-(x_t) + 1)}$.

• **The merging of Cl^+ and Cl^- .**

One needs to compute $h_t(\cdot) := h_t(\cdot)_{|Cl^+} \wedge h_t(\cdot)_{|Cl^-}$.

◊ For $t = n$ and considering that $h_n(\cdot)$ is a multi-linear combination of the α_i appearing in $h_n(\cdot)_{|Cl^+}$ or $h_n(\cdot)_{|Cl^-}$, we have that :

$$\begin{aligned}
\text{len}(h_n) &\leq 2^{\#(V^+(x_n) \cup V^-(x_n)) + 1} \\
&\leq 2^{\#V(x_n) + 1}
\end{aligned} \tag{22}$$

◊ For $t = n - 1$ also, $h_{n-1}(\cdot)$ is a multi-linear combination of the α_i corresponding to the variables x_i ($i < n - 1$) found in common clauses with x_{n-1} as the highest variable : $\text{len}(h_{n-1}) \leq 2^{\#V(x_{n-1}) + 1}$. **But** one has **to add** the potential α_i involved in the recursive

call $g_{n-1}^*(\cdot)$ from the previous computation of $h_n(\cdot)$. [see (16)]

From (16), $g_{n-1}^*(\alpha_1, \dots, \alpha_{n-1}) = [h_n(\cdot, 0)_{|Cl+} + h_n(\cdot, 0)_{|Cl-}] \cdot [h_n(\cdot, 1)_{|Cl+} + h_n(\cdot, 1)_{|Cl-}]$. So $g_{n-1}^*(\cdot)$ will be a multi-linear combination of the same α_i , except α_n , as for $h_n(\cdot)$. Therefore, $h_{n-1}(\cdot) \wedge g_{n-1}^*(\cdot)$ will be a combination of the α_i associated to the variables in $\bigcup_{i=n-1}^n [V^+(x_i) \cup V^-(x_i)] \Rightarrow \text{len}(h_{n-1} \wedge g_{n-1}^*) \leq 2^{\#\bigcup_{i=n-1}^n V(x_i)}$, as x_{n-1} should not be counted in $V(x_n)$.

◇ So, $\forall t : \text{len}(h_t \wedge g_t^*) \leq 2^{\#\bigcup_{i=t}^n V(x_i)} + 1 - (n-t)$. But as $h_t \wedge g_t^*$ is a combination of at most t α_i 's, we have :

$$\text{len}(h_t \wedge g_t^*) \leq \min(2^{\#\bigcup_{i=t}^n V(x_i)} + 1 - (n-t) , 2^t) \quad (23)$$

• Uniform distribution of the literals

As $V(x_i)$ is dependent of the ordering of the variables, it is possible to re-order the variables so that $2^{\#\bigcup_{i=t}^n V(x_i)}$ is minimal, except in the case of uniformly distributed literals. **The uniformly distributed case is then the most difficult problem**, as no ordering can reduce the maximum value in (23). ■

Example of non uniformly distributed 3-CNF-SAT problem :

- Consider the following 3-CNF-SAT problem with m clauses and $2m+1$ propositional variables :

$$\varphi := \bigwedge_{1 \leq i \leq m} (x_{2i-1} \vee x_{2i} \vee x_{2m+1})$$

Here we have :

$$V^+(x_t) = V^-(x_t) = \emptyset \quad \forall t \neq 2m+1$$

$$V^+(x_{2m+1}) = \{x_1, \dots, x_{2m}\} \quad \text{and} \quad V^-(x_{2m+1}) = \emptyset$$

$$\max_{1 \leq t \leq n} \max_{1 \leq j \leq m} \text{len}_j(h_t) = 3^{m+1} - 3^m + 1 = \mathcal{O}(3^m)$$

Note : The proof of this equality is more difficult than interesting, so we do not write it here.

- But the *same* 3-CNF-SAT problem can be formalized in terms of opposite literals

$y_i = \neg x_i, \forall i \in \{1, \dots, 2m+1\}$:

$$\varphi := \bigwedge_{1 \leq i \leq m} (\neg y_{2i-1} \vee \neg y_{2i} \vee \neg y_{2m+1})$$

This time, we have :

$$V^+(y_t) = V^-(y_t) = \emptyset \quad \forall t \neq 2m+1$$

$$V^+(y_{2m+1}) = \emptyset \quad \text{and} \quad V^-(y_{2m+1}) = \{y_1, \dots, y_{2m}\}$$

$$\max_{1 \leq t \leq n} \max_{1 \leq j \leq m} \text{len}_j(h_t) = 2^m = \mathcal{O}(2^m)$$

$$\text{Indeed, } h_t(\alpha_1, \dots, \alpha_t) = \alpha_t \quad \forall t < 2m+1$$

$$\begin{aligned} \text{and } h_{2m+1}(\alpha_1, \dots, \alpha_{2m+1}) &= [(\alpha_1 \alpha_2 \alpha_{2m+1} + \alpha_{2m+1}) \wedge (\alpha_3 \alpha_4 \alpha_{2m+1} + \alpha_{2m+1})] \wedge \dots \\ &\stackrel{\text{see(15)}}{=} [(\alpha_{2m+1} + 1) \cdot 0 + (\alpha_{2m+1}) \cdot (\alpha_1 \alpha_2 + 1) \cdot (\alpha_3 \alpha_4 + 1)] \wedge \dots \\ &= (\alpha_{2m+1}) \cdot \prod_{i=1}^m (\alpha_{2i-1} \alpha_{2i} + 1) \\ &\Rightarrow \text{len}_{2m+1}(h_{2m+1}) = 2^m \end{aligned}$$

- Finally, the *same* 3-CNF-SAT problem can be formalized using re-ordered propositional variables $z_1 = y_{2m+1}$ and $z_i = y_{i-1}, \forall i \in \{2, \dots, 2m+1\}$:

$$\varphi := \bigwedge_{1 \leq i \leq m} (\neg z_{2i} \vee \neg z_{2i+1} \vee \neg z_1)$$

And, this time, we have :

$$\begin{aligned} V^+(z_t) &= \emptyset \quad \forall t \\ V^-(z_t) &= \begin{cases} \emptyset & \text{for } t = 1 \text{ or } t = 2i \ (1 \leq i \leq m) \\ \{z_1, z_{2i}\} & \text{for } t = 2i+1 \ (1 \leq i \leq m) \end{cases} \\ \max_{1 \leq t \leq n} \max_{1 \leq j \leq m} \text{len}_j(h_t) &= 2 \end{aligned}$$

So for this example, one can reach a **linear complexity** of $\mathcal{O}(2 \cdot \text{number of } h_t(\cdot)) = \mathcal{O}(2m)$, as only one $h_t(\cdot)$ has to be computed at each step without any recursive call.

Example of uniformly distributed 3-CNF-SAT problem :

The smallest exact uniformly distributed and optimally re-ordered 3-CNF-SAT problem is :

$$\varphi := \bigwedge_{i=1}^8 \psi_i = \bigwedge \left\{ \begin{array}{l} x_1 \vee \neg x_2 \vee \neg x_3 \\ x_1 \vee x_2 \vee \neg x_3 \\ \neg x_1 \vee \neg x_2 \vee \neg x_3 \\ \neg x_1 \vee x_2 \vee \neg x_3 \\ x_1 \vee \neg x_2 \vee x_3 \\ x_1 \vee x_2 \vee x_3 \\ \neg x_1 \vee \neg x_2 \vee x_3 \\ \neg x_1 \vee x_2 \vee x_3 \end{array} \right.$$

No relabeling will reduced the 3-CNF-SAT complexity. This problem is “*hard*” in the sense that each clause eliminates only one solution at a time. We have here :

$$V^+(x_1) = V^-(x_1) = \emptyset$$

$$V^+(x_2) = V^-(x_2) = \emptyset$$

$$V^+(x_3) = V^-(x_3) = \{x_1, x_2\}$$

and

| Step | $h_1(\cdot)$ | $h_2(\cdot)$ | $h_3(\cdot)$ | $\max_t \text{len}(h_t)$ | #Solutions |
|----------------------------|--------------|--------------------|---|--------------------------|------------|
| ψ_1 | α_1 | α_2 | $(\alpha_1 + 1)\alpha_2\alpha_3 + \alpha_3$ | 3 | 7 |
| $\psi_1 \wedge \psi_2$ | α_1 | α_2 | $\alpha_1\alpha_3$ | 1 | 6 |
| $\bigwedge_{i=1}^3 \psi_i$ | α_1 | α_2 | $\alpha_1\alpha_2\alpha_3 + \alpha_1\alpha_3$ | 2 | 5 |
| $\bigwedge_{i=1}^4 \psi_i$ | α_1 | α_2 | 0 | 1 | 4 |
| $\bigwedge_{i=1}^5 \psi_i$ | α_1 | $\alpha_1\alpha_2$ | 0 | 1 | 3 |
| $\bigwedge_{i=1}^6 \psi_i$ | 1 | α_2 | 0 | 1 | 2 |
| $\bigwedge_{i=1}^7 \psi_i$ | 1 | 0 | 0 | 1 | 1 |
| $\bigwedge_{i=1}^8 \psi_i$ | \nexists | \nexists | \nexists | 0 | 0 |

Conclusions :

The theorem about uniformity is important as it states : for any non uniformly distributed 3-CNF-SAT problem φ with m clauses and n variables, there exists an uniformly distributed 3-CNF-SAT problem φ' with m clauses and n variables which is more difficult to solve, in terms of functional descriptors.

C. A sorting algorithm to reduce complexity

We propose the following sorting algorithm of complexity $\mathcal{O}(m + n \log(n) + m \log(m))$:

- ◊ Relabel the propositional variables $[x_i \rightarrow y_j]$ in order to get their occurrence $[\mathcal{O}(m)]$ in a decreasing order : $\#\{y_1\}$ is maximal, \dots , $\#\{y_n\}$ is minimal $[\mathcal{O}(n \log(n))]$;
- ◊ Inverse the sign of the literals in order to get the maximum of negative literals;
- ◊ Sort the clauses to get a increasing order of the highest variable in the ordered clauses $[\mathcal{O}(m \log(m))]$;
- ◊ Within the set of clauses with the same highest variable, sort the clauses so that the ones with negative highest variable appear before the ones with positive highest variable.

As we seldom have exact uniformly distributed 3-CNF-SAT problems, the complexity can then be reduced drastically, as shown in Figure 1.

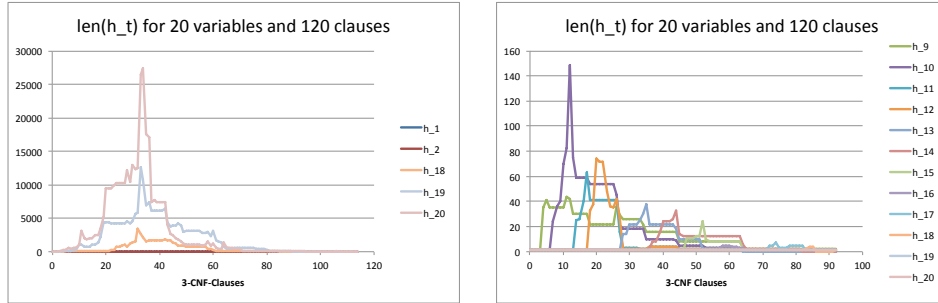


Fig. 1 : Complexity for the same dataset before and after the sorting algorithm.

D. Exact uniformly distributed α -random 3-CNF-SAT problems

Definition : Let φ be a 3-CNF-SAT problem with n variables, each of them appearing exactly $\frac{3\alpha}{2}$ times as positive and $\frac{3\alpha}{2}$ times as negative literal, for some $\alpha > 0$. Let these $3\alpha n$ literals be randomly distributed amongst the clauses. Such problem is called ***an exact uniformly distributed α -random 3-CNF-SAT problem.***

Remark : For exact uniformly distributed 3-CNF-SAT problem, the labeling part of the previous *sorting algorithm* has no effect, as the occurrence of each literal is $\frac{3\alpha}{2}$. Only the re-ordering of the m clauses can reduce the complexity of the problem.

Theorem VI.3: For such exact uniformly distributed α -random 3-CNF-SAT problems, the expected number of clauses where i ($i > 2$) is the highest index, is noted $m_\alpha(i)$ and given by :

$$E[\#\{\psi = [\neg]x_r \vee [\neg]x_s \vee [\neg]x_t | \max(r, s, t) = i\}] = \frac{(i-1)(i-2)}{(n-1)(n-2)} 3\alpha \equiv m_\alpha(i) \quad (24)$$

Proof: Let ψ be a clause with x_i or $\neg x_i$, there are $C_2^{i-1} \cdot 3\alpha$ combinations with smaller indices amongst $C_2^{n-1} \cdot 3\alpha$ possibles combinations. So, the probability for x_i to get the highest index is : $\frac{(i-1)(i-2)}{(n-1)(n-2)}$. The expected value is obtained by multiplying the probability by the number of occurrences of x_i . ■

Figure 2 shows the theoretical density and cumulative distributions of $m_\alpha(i)$ versus the distributions for the observed values for $\#\{\psi = [\neg]x_r \vee [\neg]x_s \vee [\neg]x_t | \max(r, s, t) = i\}$ in the case of a 3-CNF-SAT problem with 175 variables and 753 random clauses.

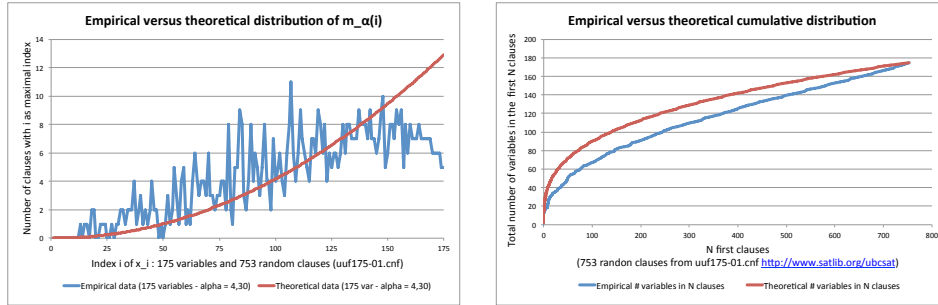


Fig. 2 : Density and cumulative distributions of “sorted clauses” for $n = 175$ and $\alpha = 4,30$.

Theorem VI.4: For exact uniformly distributed α -random 3-CNF-SAT problems, the expected number of variables in $V(x_i)$, for $i > 2$ and large n , is given by :

$$E[\#V(x_i)] = 2 m_\alpha(i) = \frac{(i-1)(i-2)}{(n-1)(n-2)} 6\alpha \quad (25)$$

Proof: There is C_2^{i-1} possible triplets with x_i being the highest indexed variable. The probability for some x_j ($j < i$) to appear in one of these triplets is $\frac{i-2}{C_2^{i-1}} = \frac{2}{i-1} = p$ for any j . The occurrence of x_j follows a binomial model $Bi(m_\alpha(i), p)$, as one can choose several

times the same triplet (given different clauses with respect to the negative or positive sign of the included literals). The expected number of occurrence of x_j in the $m_\alpha(i)$ triplets is then $m_\alpha(i) \cdot p = \frac{6\alpha(i-2)}{(n-1)(n-2)} < 1$ for large n . So each variable is expected to appear at most once in the $m_\alpha(i)$ triplets-clauses. Therefore, the number of variables, different from x_i , occurring in these $m_\alpha(i)$ clauses is $2m_\alpha(i)$ as there are two variables distinct from x_i in each clause. ■

Theorem VI.5: For exact uniformly distributed α -random 3-CNF-SAT problems, the maximal expected complexity for the computation of $h_t(\cdot)$ is bounded by :

$$\text{len}(h_t) \leq \max_{k \geq 0} 2 \left[\min \left\{ 2 \left(\sum_{j=0}^k m_\alpha(n^{(j)}) \right) - (k-1), n^{(k)} \right\} \right]$$

where $2 \left(\sum_{j=0}^k m_\alpha(n^{(j)}) \right) - (k-1)$ is a concave quadratic function with respect to t or k , as shown on figures 3 and 4.

Proof: From (25), we know that $\#V(x_i)$ is expected to be maximal for $i = n$, when $\#V(x_n) = 2m_\alpha(n) = 6\alpha$. So, from (22), we have that :

$$\begin{aligned} \text{len}(h_n) &\leq 2^{\#V(x_n) + 1} \\ &\leq 2^{2m_\alpha(n) + 1} = 2^{(6\alpha + 1)} \end{aligned} \tag{26}$$

For the computation of the recursive call $g_j^*(\alpha_1, \dots, \alpha_j)$ (see 16), the index j is the highest index of the variables in $V(x_n)$. Let us note it $n^{(1)}$. We have thus $\#V(x_n) = 2m_\alpha(n)$ indexes uniformly chosen from $\{1, \dots, n-1\}$. $n^{(1)}$ will be the expected maximal index from an uniform distribution for $2m_\alpha(n)$ iid variables $u_i \sim U[1, \dots, n-1]$:

$$n^{(1)} = E \left[\max_{1 \leq i \leq 2m_\alpha(n)} (u_i) \right] = \frac{2m_\alpha(n)}{2m_\alpha(n) + 1} (n-1) = \frac{6\alpha}{6\alpha + 1} (n-1) \tag{27}$$

So, for the recursive call, we will have to compute $h_{n^{(1)}}(\cdot) \wedge g_{n^{(1)}}^*(\cdot)$. We get :

$$\#V(x_{n^{(1)}}) = 2 m_\alpha(n^{(1)}) = \frac{(n^{(1)} - 1)(n^{(1)} - 2)}{(n - 1)(n - 2)} 6 \alpha \quad \text{from (25)}$$

$$\begin{aligned} \text{len}(h_{n^{(1)}}(\cdot) \wedge g_{n^{(1)}}^*(\cdot)) &\leq 2^{\#\{V(x_n) \cup V(x_{n^{(1)}})\} + 1 - (2-1)} \quad \text{from (23)} \\ &\leq 2^{2(m_\alpha(n) + m_\alpha(n^{(1)}))} \end{aligned}$$

And so on, for the next recursive calls. We get for the recursive call k ($k > 1$) :

$$n \equiv n^{(0)}$$

$$u_i \sim U[1, \dots, n^{(k-1)} - 1]$$

$$n^{(k)} = E\left[\max_{1 \leq i \leq 2m_\alpha(n^{(k-1)})} (u_i)\right] = \frac{2 m_\alpha(n^{(k-1)})}{2 m_\alpha(n^{(k-1)}) + 1} (n^{(k-1)} - 1)$$

$$\#\{V(x_{n^{(k)}})\} = 2 m_\alpha(n^{(k)}) = \frac{(n^{(k)} - 1)(n^{(k)} - 2)}{(n - 1)(n - 2)} 6 \alpha$$

$$\begin{aligned} \text{len}(h_{n^{(k)}}(\cdot) \wedge g_{n^{(k)}}^*(\cdot)) &\leq 2^{\left[\min\left\{ \# \bigcup_{j=0}^k \{V(x_{n^{(j)}})\} - (k - 1), n^{(k)} \right\} \right]} \\ &\leq 2^{\left[\min\left\{ 2 \left(\sum_{j=0}^k m_\alpha(n^{(j)}) \right) - (k - 1), n^{(k)} \right\} \right]} \\ &\leq 2^{\left[\min\{M_\alpha(n^{(k)}), n^{(k)}\} \right]} \end{aligned} \tag{28}$$

This bound is only defined for the variables with $n^{(k)}$ as index. Note that $n^{(k)}$ are functions of the starting index $n^{(0)} = n$. We can compute similar bounds for other starting indexes $n^{(0)}$ in $[1, \dots, n - 1]$, so that $M_\alpha(\cdot)$ can be defined for all t as shown in Figure 4.

Numerical computations show that, for large n , $M_\alpha(n^{(k)})$ as well as $M_\alpha(t)$ are concave quadratic functions with coefficients only depending on α . This can be easily explained as a mere consequence of the *iid* randomness of the variables $\#\{V(x_{n^{(k)}})\}$ and $m_\alpha(n^{(j)})$.

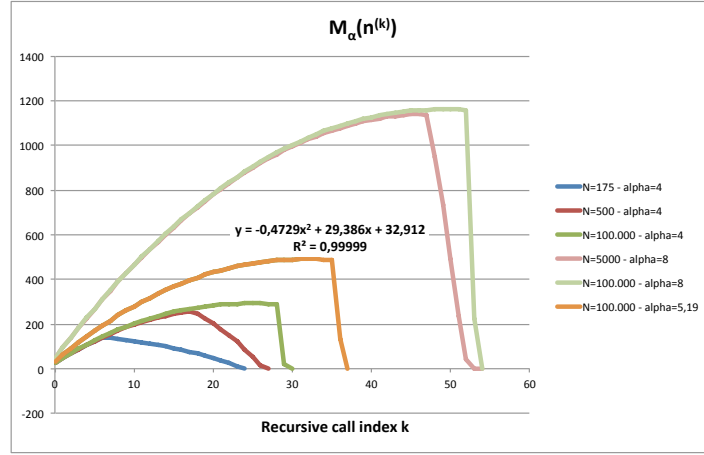


Fig. 3 : Complexity wrt k : $M_\alpha(n^{(k)}) = 2(\sum_{j=0}^k m_\alpha(n^{(j)})) - (k-1)$ where $n^{(0)} = n$.

Indeed, the *central limit theorem* for the expectation of *iid* random variables predicts that $E[2^{M_\alpha(n^{(k)})}]$ follows a Normal distribution (censored by min). But $X \sim N(\mu, \sigma^2)$ implies a quadratic log-density : $\log(f_X(x)) \propto -\frac{(x-\mu)^2}{2\sigma^2}$. For each value of α , we can compute the corresponding μ_α , σ_α and the maximum value for $M_\alpha(n^{(k)})$. Quadratic regression estimations give $\max_k(M_\alpha(n^{(k)})) \approx 294$ for $\alpha = 4$, $\max_k(M_\alpha(n^{(k)})) \approx 490$ for $\alpha = 5, 12$ (see figure 3 and below for the choice of such α) and $\max_t(M_\alpha(t)) \approx 1160$ for $\alpha = 8$.

Remark : It is now important to see whether different starting points $n^{(0)}$ yield not to aggregating trajectories so that addition of bounds are to be considered. This situation can be neglected as shown in the following theorem. ■

Theorem VI.6: The probability for a given variable x_i to be in more than one trajectory tends to zero for large n .

Proof: Let us consider separately the possible trajectories $tr(x_{n^{(0)}} \rightarrow x_{n^{(k)}})$ for $n^{(0)} = m \in \{1, \dots, n\}$ and $k \in \{1, \dots, n\}$. Let us note a given trajectory : $tr(m, k_m)$ with k_m such that $n^{(k_m)} > i$. For each variable x_i , there exists at most $(n-i)(n-i-1)/2$ trajectories $tr(m, k_m)$ where x_i could be the next highest indexed variable for $n^{(k_m+1)}$: $tr(n, 0), \dots, tr(n, k_n), \dots$,

$tr(i+1, 0)$. The probability for x_i to get the highest index in a trajectory is :

$$\begin{aligned}
 P[i = \max_l \{l : x_l \in tr(m, k_m)\}] \\
 &= P[i = \max_l \{l : x_l \in \bigcup_{j=0}^{j=k_m} V(x_{n^{(j)}}) \setminus \{x_{n^{(0)}}, \dots, x_{n^{(k_m)}}\}\}] \\
 &= P[i = \max_l \{l : x_l \in tr(m, k_m)\} | x_i \in tr(m, k_m)] \cdot P[x_i \in tr(m, k_m)]
 \end{aligned}$$

We have :

$$\begin{aligned}
 P[x_i \in tr(m, k_m)] &= \#\{\text{clauses in } tr(m, k_m)\} \cdot P[x_i \text{ in the clause and } i \text{ is the maximum index}] \\
 &= \sum_{j=0}^{j=k_m} m_\alpha(n^{(j)}) \cdot \frac{n^{(j)} - 2}{C_2^{n^{(j)} - 1}} \quad [\text{see (24)}] \\
 &= \sum_{j=0}^{j=k_m} m_\alpha(n^{(j)}) \cdot \frac{2}{n^{(j)} - 1} \\
 &= \sum_{j=0}^{j=k_m} \frac{(n^{(j)} - 1)(n^{(j)} - 2)}{(n - 1)(n - 2)(n^{(j)} - 1)} 6\alpha \\
 &= \frac{(6\alpha)}{(n - 1)(n - 2)} \sum_{j=0}^{j=k_m} (n^{(j)} - 2)
 \end{aligned}$$

For instance :

$$P[x_i \in tr(m, 0)] = \frac{6\alpha(n^{(0)} - 2)}{(n - 1)(n - 2)} = \frac{6\alpha}{n - 1} \frac{m - 2}{n - 2} \leq \frac{6\alpha}{n - 1} \text{ as } n^{(0)} = m \leq n$$

and

$$\begin{aligned}
 P[x_i \in tr(m, 1)] &\leq P[x_i \in tr(n, 1)] \\
 &= \frac{6\alpha}{(n - 1)(n - 2)} (n^{(0)} - 2) + (n^{(1)} - 2) \\
 &= \frac{6\alpha}{n - 1} + \frac{6\alpha}{(n - 1)(n - 2)} \left(\left[\frac{6\alpha}{6\alpha + 1} (n - 1) \right] - 2 \right) \\
 &< \frac{6\alpha}{n - 1} + \frac{6\alpha}{(n - 1)(n - 2)} (n - 3) \\
 &< \frac{6\alpha}{n - 1} \left(1 + \frac{n - 3}{n - 2} \right)
 \end{aligned}$$

Finally,

$$\begin{aligned}
 P[x_i \in tr(m, k_m)] &\leq P[x_i \in tr(n, k_m)] < \frac{6\alpha}{n - 1} \left(1 + \frac{n - 3}{n - 2} + \dots + \frac{n - (k_m + 2)}{n - 2} \right) \\
 &\rightarrow 0 \text{ for large } n \text{ with respect to } k_m \text{ and } \alpha.
 \end{aligned}$$

Now, considering that the elements of $tr(m, k_m)$ are *iid* uniformly distributed random variables drawn from $\{1, \dots, m-1\}$, we have :

$$\begin{aligned}
P[i = \max_l \{l : x_l \in tr(m, k_m)\} | x_i \in tr(m, k_m)] \\
&= P[i = \max\{\#tr(m, k_m) \text{ uniform random variables}\}] \\
&\quad [\text{for large } n, \text{ we use the expected value for } \#tr(m, k_m)] \\
&= P[\sum_{j=0}^{j=k_m} 2m_\alpha(n^{(j)}) - 1 \text{ uniform iid variables } \leq i] \\
&= \prod_{l=1}^{\sum_{j=0}^{j=k_m} 2m_\alpha(n^{(j)}) - 1} \frac{i}{(m-1)} \\
&= \left(\frac{i}{m-1}\right)^{\sum_{j=0}^{j=k_m} 2m_\alpha(n^{(j)}) - 1}
\end{aligned}$$

In conclusion,

$$\begin{aligned}
P[i = \max_l \{l : x_l \in tr(m, k_m)\}] &= \left(\frac{i}{m-1}\right)^{\sum_{j=0}^{j=k_m} 2m_\alpha(n^{(j)}) - 1} \frac{(6\alpha)}{(n-1)(n-2)} \sum_{j=0}^{j=k_m} (n^{(j)} - 2) \\
&\leq \frac{(6\alpha)}{(n-1)(n-2)} \sum_{j=0}^{j=k_m} (n^{(j)} - 2) \quad \text{as } i \leq (m-1) \\
&\rightarrow 0 \quad \text{for large } n \text{ with respect to } k \text{ and } \alpha
\end{aligned}$$

Therefore, there is a negligible probability for a variable x_i to be maximal in two or more trajectories $tr(m, k_m)$, as we can see this event as the output of a binomial model with a very small probability of success (" x_i being maximal in some $tr(m, k_m)$ "), over $(n-i)(n-i-1)/2$ possible trajectories :

$$\text{Let } p = \max_{m, k_m} P[i = \max_l \{l : x_l \in tr(m, k_m)\}]$$

Then, $P[\text{Two or more successes}]$

$$\begin{aligned}
&= 1 - (P[0 \text{ success}] + P[1 \text{ success}]) \\
&\leq 1 - ([(1-p)^{\frac{(n-i)(n-i-1)}{2}}] + [\frac{(n-i)(n-i-1)}{2} p(1-p)^{\frac{(n-i)(n-i-1)}{2} - 1}]) \\
&\rightarrow 0 \quad \text{for large } n, \text{ as } p \rightarrow 0 \text{ for large } n \text{ with respect to } k \text{ and } \alpha.
\end{aligned}$$

The last thing to prove is that k is not $\mathcal{O}(n)$ as α is a given constant. Figure 3, which is computed with the theoretical formula from (28), shows that the maximal value for k is

negligible with respect to n : $k \leq 30$ for $n = 100.000$ and $\alpha = 4$, and $k \leq 55$ when $n = 100.000$ and $\alpha = 8$. ■

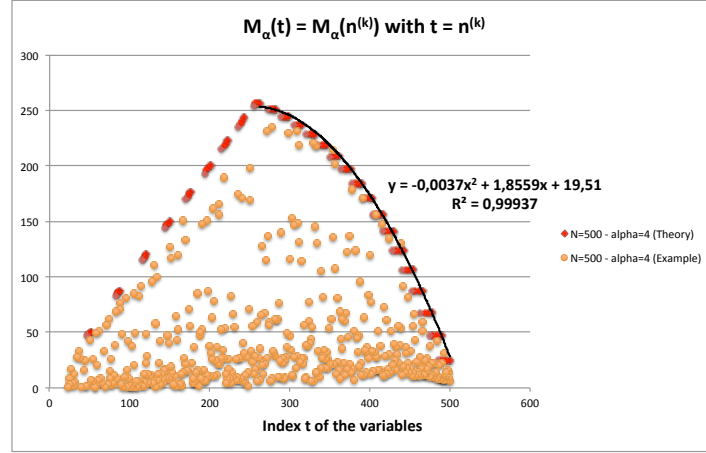


Fig. 4 : Complexity wrt variable index t : $M_\alpha(n^{(k)}) = M_\alpha(t)$ for $n^{(0)} = n$.

The example comes from the Dimacs generator at <https://toughsat.appspot.com/>

Remark : With some real generated 3-CNF-SAT problems, it is possible to observe a “cluster” process, the size of one trajectory, i.e. the number of x_i involved in that trajectory, becoming more and more important so that this trajectory attracts all the variables. Then, k is $O(n)$, $P[x_i \in tr(n, k)] \rightarrow 1$ and the complexity becomes exponential. It is easy to solve these cases. As the variables are uniformly distributed in random 3-CNF-SAT problems, each variable x_i being repeated approximatively 3α times, it is possible to permute joining variables x_j (belonging to two or more trajectories) with a smaller indexed variable, such as x_{j-1} (or x_{j-2} if x_{j-1} is already in a previous trajectory, and so on). The two trajectories will then be dissociated. We propose the following “permutation” algorithm :

- ◇ First, apply the sorting algorithm over the 3-CNF-SAT problem;
- ◇ Sort each clause $[\neg]x_r \vee [\neg]x_s \vee [\neg]x_t$ so that $r \geq s \geq t$;
- ◇ Beginning with the last clause (with $[\neg]x_n$), mark x_j where $j = \max\{i : x_i \in V(x_n)\}$ as already belonging in a trajectory and initialize $W(x_n) := V(x_n)$ and $W(x_j) := V(x_n)$ where $W(x_j) \equiv \cup_i V(x_i)$ for i such that $x_i \in tr(\dots, x_j)$;
- ◇ Loop over $k := 1$ to $k := n - 3$ with clauses having $[\neg]x_{n-k}$ as the highest indexed variable; if x_{n-k} is already marked as belonging to a trajectory, do $W(x_{n-k}) := W(x_{n-k}) \cup V(x_{n-k})$ otherwise initialize $W(x_{n-k}) := V(x_{n-k})$;

Consider x_j where $j = \max\{i : x_i \in W(x_{n-k})\}$; do while (x_j is already marked as belonging in a trajectory and $j \geq 3\alpha$) relabel $x_j \leftrightarrow x_{j-1}$ and $j := j - 1$

[we do not consider $j < 3\alpha$ as merging of trajectories for small indexes is not a problem because $M_\alpha(j) = j$];

Initialize $W(x_j) := W(x_{n-k})$.

Figure 4 shows the result for a Dimacs generated 3-CNF-SAT problem with 500 variables and $\alpha = 4$. We apply the sorting and the permuting algorithms on the generated file to eliminate joining trajectories.

If we have proved in this section that the complexity is bounded with respect to n , we still have to show that complexity is not increasing with respect to α , which is not the case for $M_\alpha(t)$.

VII. COMPLEXITY ANALYSIS WITH RESPECT TO α

It is easy to see that the complexity is an increasing function of α for exact uniformly distributed α -random 3-CNF-SAT problems, at least for small α , as smaller α -random 3-CNF-SAT problems can be viewed as subsets of larger α -random problems.

But there should be somewhere **a threshold for** α as large α -random problems are easy to solve because unsatisfiability is often a consequence of a subset of the problem. Empirical results from the literature suggest that this threshold for α is ≈ 4.258 . See [3].

The analysis of complexity with respect to α will be done through \mathcal{S}_φ , the set of all satisfying solutions for the 3-CNF-SAT problem φ . See definition (2).

Theorem VII.1: For exact uniformly distributed α -random 3-CNF-SAT problems $\varphi = \{\psi_j\}_{1 \leq j \leq m}$ with n variables and m clauses, we get for large n and m the following expected number of solutions :

$$E[\#\mathcal{S}_\varphi] = E[\#\{(x_1, \dots, x_n) \in \{0, 1\}^n \mid \varphi(x_1, \dots, x_n) = 1\}] = 7 \left(\frac{7}{4}\right)^{n-3} \left(\frac{7}{8}\right)^{(m-n+2)} \quad (29)$$

Proof:

- Let us *re-order* the m clauses ψ_j in such a way that each clause has only *one* new additional variables with respect to the set of variables appearing in the previous clauses.
- Let $\mathcal{V}_k = \{x_i \mid \exists j, 1 \leq j \leq k : x_i \text{ appears in } \psi_j\}$. The *re-ordering* of the clauses yields to embedded subsets $\mathcal{V}_1 \subseteq \mathcal{V}_2 \subseteq \dots \subseteq \mathcal{V}_{n-2} = \{x_1, \dots, x_n\}$ with $\#\mathcal{V}_1 = 3, \dots, \#\mathcal{V}_k = k+2, \dots, \#\mathcal{V}_{n-2} = n$ and $\#\mathcal{V}_{k'} = n \ \forall k' \geq n-2$.

The cases where all clause ψ_{k+1} introduces *two or three* new additional variables to \mathcal{V}_k are to be neglected, as this means that the 3-CNF-SAT problem can be split into two sub-problems with one or zero common variable, which reduces drastically the complexity of the problem.

- Let us look at the expected effect of a clause ψ_j ($1 \leq j \leq m$) over the number of solutions :

1. Let us consider ψ_1 .

The first clause yields to $7 \cdot 2^{n-3}$ solutions. The *matrix representation* of ψ_1 will be a 7×3 matrix.

2. Let us consider ψ_2 .

Let ψ_2 introduces only one new additional variable x_t , and let x_r and x_s be the two common variables for ψ_1 and ψ_2 :

$$[\psi_1] = \left(\begin{array}{ccc} x_q & x_r & x_s \\ \hline & 7 \text{ lines} & \end{array} \right) \text{ and } [\psi_2] = \left(\begin{array}{ccc} x_r & x_s & x_t \\ \hline & 7 \text{ lines} & \end{array} \right)$$

Depending on the sign of the literal x_t in ψ_2 , the result matrix for $[\psi_1 \wedge \psi_2]$ will get the 7 lines of $[\psi_1]$ with a *zero* in the column for x_t if $\psi_2 = [\neg]x_r \vee [\neg]x_s \vee \neg x_t$ or with a *one* if $\psi_2 = [\neg]x_r \vee [\neg]x_s \vee x_t$. This corresponds to solutions where the literal $[\neg]x_t$ is satisfied.

On the contrary, when the value in the column for x_t is opposite to the sign of $[\neg]x_t$, the satisfiability of ψ_2 should pass through the literals x_r and x_s . Among the four possible values for (x_r, x_s) , only three will be accepted. One couple for (x_r, x_s) will be ruled out, as well in matrix $[\psi_1]$ as in $[\psi_2]$. This corresponds to one or two lines deleted in $[\psi_1]$, depending on the sign for x_r and x_s in ψ_1 . The expected number of lines deleted in $[\psi_1]$ will be : $1 \cdot P[\text{one deletion}] + 2 \cdot P[\text{two deletions}] = 1 \cdot \frac{1}{4} + 2 \cdot \frac{3}{4} = \frac{7}{4}$. Therefore, the expected number of lines in $[\psi_1 \wedge \psi_2]$ will be equal to $7 + (7 - \frac{7}{4}) = 7(1 + \frac{3}{4}) = 7(\frac{7}{4}) = 12,25$. And the boundaries for $\#[\psi_1 \wedge \psi_2]$

are $[\min_2, \max_2] = [12, 13]$.

3. Let us consider ψ_3 .

Let ψ_3 introduce a new additional variable. Using the same type of arguments as for ψ_2 , the expected number of deleted lines in $[\psi_1 \wedge \psi_2]$ will be equal to :

$$E(\# \text{ deletions in } \psi_1 \wedge \psi_2) = \sum_{k=12}^{13} \left(\sum_{d=1}^{2^2} d \cdot P[d \text{ deletions } | \#[\psi_1 \wedge \psi_2] = k] \right) \cdot P[\#[\psi_1 \wedge \psi_2] = k]$$

Let us consider here an example where $\#[\psi_1 \wedge \psi_2] = 13$.

$$\text{For instance, } [\varphi] = [(x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee \neg x_3 \vee \neg x_4)] = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Let us consider the couples (x_i, x_j) and the number of deleted lines for each case :

| x_i | x_j | # del. | | x_1 | x_2 | # del. |
|-------|-------|--------|-------------------------|-------|-------|--------|
| 0 | 0 | d_1 | | 0 | 0 | 2 |
| 0 | 1 | d_2 | For the above example : | 0 | 1 | 4 |
| 1 | 0 | d_3 | | 1 | 0 | 3 |
| 1 | 1 | d_4 | | 1 | 1 | 4 |

We see that, whatever the value of $\#[\psi_1 \wedge \psi_2]$, $\sum_{i=1}^4 d_i = \#[\psi_1 \wedge \psi_2]$. So, the expected number of deleted clauses, independently from the case (x_i, x_j) , will be :

$$d_1 \cdot \frac{1}{4} + d_2 \cdot \frac{1}{4} + d_3 \cdot \frac{1}{4} + d_4 \cdot \frac{1}{4} = \frac{\sum_i d_i}{4} = \frac{\#[\psi_1 \wedge \psi_2]}{4}$$

Therefore, the expected number of lines in $[\bigwedge_{i=1}^3 \psi_i]$ will be :

$$\begin{aligned}
E[\#[\bigwedge_{i=1}^3 \psi_i]] &= \sum_k \{ \#[\psi_1 \wedge \psi_2] + (\#[\psi_1 \wedge \psi_2] - \# \text{ deletions}) \} \cdot P[\#[\psi_1 \wedge \psi_2] = k] \\
&= \sum_{k=12}^{13} \{ \#[\psi_1 \wedge \psi_2] + (\#[\psi_1 \wedge \psi_2] - \frac{\#[\psi_1 \wedge \psi_2]}{4}) \} \cdot P[\#[\psi_1 \wedge \psi_2] = k] \\
&= \left(12 \{ 1 + (1 - \frac{1}{4}) \} \cdot \frac{3}{4} \right) + \left(13 \{ 1 + (1 - \frac{1}{4}) \} \cdot \frac{1}{4} \right) \\
&= E[\#[\bigwedge_{i=1}^2 \psi_i]] \cdot \frac{7}{4} \\
&= 7 \cdot \left(\frac{7}{4} \right)^2 \\
&= 21,4375
\end{aligned}$$

And $\#[\bigwedge_{i=1}^3 \psi_i] \in [\min_3, \max_3] = [12 + 12 - 4, 13 + 13 - 1] = [20, 25]$

4. Let us now consider a given clause ψ_j ($j \leq n-2$).

We know that ψ_j introduces a new additional variable. Then, using the same type of arguments as for ψ_3 , the expected number of lines will be :

$$\begin{aligned}
E[\#[\bigwedge_{i=1}^j \psi_i]] &= E[\#[\bigwedge_{i=1}^{j-1} \psi_i]] \cdot \left(\frac{7}{4} \right) = 7 \cdot \left(\frac{7}{4} \right)^{j-1}. \\
\text{and } \#[\bigwedge_{i=1}^j \psi_i] &\in [2 \min_{j-1} - 2^{j-1}, 2 \max_{j-1} - 1] \\
&\in [\max\{0, 2^{j-1}(8-j)\}, 6 \cdot 2^{j-1} + 1]
\end{aligned}$$

5. So, for ψ_{n-2} , we have ($n \geq 10$) :

$$E[\#[\bigwedge_{i=1}^{n-2} \psi_i]] = 7 \cdot \left(\frac{7}{4} \right)^{n-3} \quad (30)$$

$$\text{and } \#[\bigwedge_{i=1}^{n-2} \psi_i] \in [0, 6 \cdot 2^{n-3} + 1] \quad (31)$$

6. For ψ_j where $j > n-2$, no new variable will be added, and the number of solution will only decrease.

Using the same previous argument, we can consider the six possible cases (x_i, x_j, x_k)

and the corresponding d_i with $1 \leq i \leq 8$. Here again, we get that :

$$\sum_{i=1}^8 d_i = \#[\bigwedge_{i=1}^{j-1} \psi_i]$$

Thus, the expected number of deleted lines in $[\bigwedge_{i=1}^{j-1} \psi_i]$ will be $\frac{\#[\bigwedge_{i=1}^{j-1} \psi_i]}{8}$.

There is no other operation to do for ψ_j . We only have to delete some lines in $[\bigwedge_{i=1}^{j-1} \psi_i]$. So, computing the remaining lines, we get :

$$E[\#[\bigwedge_{i=1}^j \psi_i]] = E[\#[\bigwedge_{i=1}^{j-1} \psi_i]] \cdot \frac{7}{8}$$

And the boundaries will be :

$$\#[\bigwedge_{i=1}^j \psi_i] \in [0, \max_{j-1} - 1]$$

7. Finally, for the last clause ψ_m , we get :

$$E[\#\mathcal{S}_\varphi] = E[\#[\bigwedge_{i=1}^m \psi_i]] = 7 \left(\frac{7}{4}\right)^{n-3} \left(\frac{7}{8}\right)^{(m-n+2)}$$

$$\text{and } \#[\bigwedge_{i=1}^m \psi_i] \in [0, 6 \cdot 2^{n-3} - m + n - 1]$$

■

Theorem VII.2: *The most difficult exact uniformly distributed α -random 3-CNF-SAT problems are the ones with a ratio $\alpha = \frac{m}{n}$ approximately equal to 5,19.*

Proof: Let us consider *exact uniformly distributed α -random 3-CNF-SAT problems*. The most difficult problems are the ones where the decision between satisfiability and unsatisfiability arises only when considering the last clause ψ_m . This is equivalent to have $E[\#\mathcal{S}_\varphi] \approx 1$.

We get :

$$\begin{aligned} E[\#\mathcal{S}_\varphi] \approx 1 &\Leftrightarrow 7 \left(\frac{7}{4}\right)^{n-3} \left(\frac{7}{8}\right)^{m-n+2} \approx 1 \\ &\Leftrightarrow \left(\frac{7}{8}\right)^m 2^n \approx 1 \\ &\Leftrightarrow m \log\left(\frac{7}{8}\right) + n \log(2) \approx 0 \\ &\Leftrightarrow (\alpha n) \log\left(\frac{7}{8}\right) + n \log(2) \approx 0 \\ &\Leftrightarrow \alpha \log\left(\frac{7}{8}\right) \approx -\log(2) \\ &\Leftrightarrow \alpha \approx \frac{-\log(2)}{\log\left(\frac{7}{8}\right)} \\ &\Leftrightarrow \alpha \approx 5,19089307 \end{aligned}$$

■

Theorem VII.3: The relation between *exact uniformly distributed α -random 3-CNF-SAT problems* and *usual α -random 3-CNF-SAT problems* can be seen as a reduction of the ratio α through the function : $3\alpha - \sqrt{1,9098\alpha}$.

Proof:

When considering *exact uniformly distributed α -random 3-CNF-SAT problems*, each literal occurs with exactly the same frequency in the m clauses, only the combination of the literals in each clause being random. We have : $\#x_i = 3\alpha$.

But the *usual uniform α -random 3-CNF-SAT problems* are such that : $E[\#x_i] = 3\alpha$, where the variables are drawn randomly from a *multinomial* population with $P[x_i \text{ appears in a clause}] = p_i = \frac{3\alpha}{m} = \frac{3}{n}$. For large n , the number of occurrence for each variable will asymptotically follow a *Normal distribution* $N(\mu, \sigma^2)$ with $\mu = m \cdot p_i = 3\alpha$ and $\sigma^2 = m \cdot p_i(1 - p_i) \approx 3\alpha$. If we consider, *after sorting the clauses as explained in our descriptor approach*, the second half of the clauses (where $M_\alpha(t) \geq t$), we will get a *folded normal distribution* for D_i [*usual α -random 3-CNF*] = $|\#x_i - E[\#x_i]| = |\#x_i - 3\alpha|$. We have :

$$\begin{aligned} D_i &\sim |N(0, 3\alpha)| \\ E[D_i] &\approx \sigma \sqrt{\frac{2}{\pi}} \\ &\approx \sqrt{\frac{6\alpha}{\pi}} = \sqrt{1,9098\alpha} \\ \Rightarrow E[\#x_i] &\approx 3\alpha - \sqrt{1,9098\alpha} \text{ for the clauses where } M_\alpha(t) \geq t \end{aligned}$$

So, if we have $\#x_i = \alpha$ in the *exact uniformly distributed α -random 3-CNF-SAT problems*, this corresponds to an “folded” expected occurency $E[\#x_i] \approx 3\alpha - \sqrt{1,9098\alpha}$ for *usual uniform α -random 3-CNF-SAT problems*. ■

Corollary VII.1: The threshold $\alpha = 5,19$ found for exact uniformly distributed α -random 3-CNF-SAT problems corresponds approximatively to a reduced threshod

$$\alpha = \frac{E[\#x_i]}{3} = 5,19 - \frac{\sqrt{1,9 \times 5,19}}{3} = 4,14135$$

for usual uniform α -random 3-CNF-SAT problems.

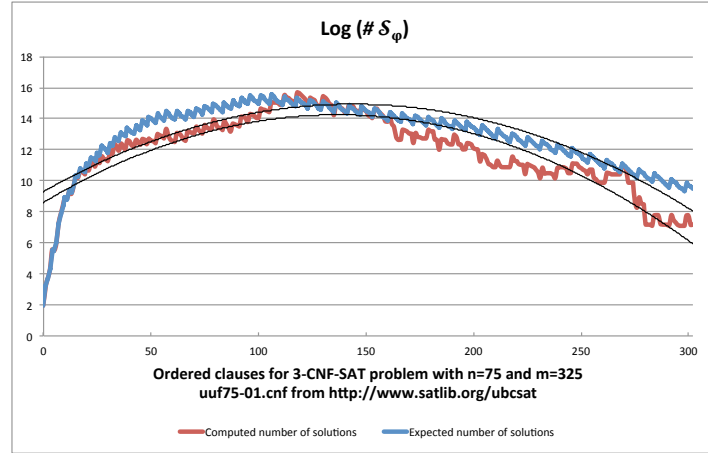


Fig. 5 : Number of solutions with respect to the analyzed clauses for a 3-CNF-SAT problem with $n = 75$ and $m = 325$

Note : This is still a theoretical value for the threshold. Indeed, for usual uniform α -random generated 3-CNF-SAT problem, we detect a small difference between the observed and the theoretical expected number of solutions with respect of the first j analyzed clauses $\bigwedge_{i=1}^j \psi_i$. The theoretical expected number of solutions $E[\#\bigwedge_{i=1}^j \psi_i]$ is defined as $7 \cdot (\frac{7}{4})^s \cdot (\frac{7}{8})^t$, where s is the number of clauses in $\{\psi_2, \dots, \psi_j\}$ introducing new additional variable and t the number of remaining clauses. Figure 5 shows the situation for a 3-CNF-SAT problem with $n = 75$ and $m = 325$, taken from <http://www.satlib.org/ubcsat>.

This difference shows that theoretical expected values are over-estimating the observed values. Let us note that we *re-ordered* the m clauses ψ_j in such a way that new additional variables are appearing as lately as possible in the 3-CNF-SAT problem (in order not to reach too large numbers for $\#\mathcal{S}_\varphi$).

VIII. CONCLUSIONS AND FUTURE RESEARCHES

Our researches were built on *exact uniformly distributed α -random 3-CNF-SAT problems*. The complexity analysis was mainly done in terms of *expected value* for some characteristics. We see that these expected values are over-estimating the real values. This means that our conclusions about the most difficult value for α $[= 5, 19]$, and therefore about the maximum theoretical value for the complexity $2^{M_\alpha(t)}$ $[= 2^{490}]$ are over fitted. Future researches will try

to suppress this bias to be more accurate in our estimation of the complexity for the NP problems.

We have seen that for $\alpha \approx 5, 19$, the maximum complexity for a α -random 3-CNF-SAT problem will be around 2^{490} whatever the number of variables. **The NP problems are then not exponential but *bounded exponential* problems. This makes them belonging to P .** But even with “yottaflops” computers (10^{24} instructions by second), this can take about “1013657570878860998520660693590880992226840594269701439142496425246188692463039064879247034987638184445605903560477” centuries to solve such problems. ☺ This is not exponential, only a huge constant upper bound.

Even if this paper is mostly theoretical, each theorem was validated by extensive numerical tests. Future researches will be to improve our different algorithms implementing the descriptor approach for 3-CNF-SAT problems¹.

REFERENCES

- [1] Stanley Burris. *A Course in Universal Algebra*. Dover Pubns, City, 2012.
- [2] Th. Cormen, Ch. Leiserson, R. Rivest, and Cl. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, 2nd edition, 2001.
- [3] James M. Crawford and Larry D. Auton. Experimental results on the crossover point in random 3-sat. *Artificial Intelligence*, 81(12):31 – 57, 1996. Frontiers in Problem Solving: Phase Transitions and Complexity.
- [4] Rémon Marcel. About the impossibility to prove $p \neq np$ or $p = np$ and the pseudo-randomness in np . Published in Arxiv : <http://arxiv.org/abs/0904.0698v2>, January 2010.
- [5] M. Sipser. The History and Status of the P versus NP Question. *Proceedings of the 24th Annual Meeting ACM*, pages 603–618, 1992.

¹ I would like to thank Dr. Johan Barthélemy for his help in terms of writing and testing these algorithms, as well as the SMART department of the University of Wollongong for their welcome. Codes will be available on www.github.com